

Minimizing a sum of submodular functions

Vladimir Kolmogorov
University College London
v.kolmogorov@cs.ucl.ac.uk

Abstract

We consider the problem of minimizing a function represented as a sum of submodular terms. We assume each term allows an efficient computation of *exchange capacities*. This holds, for example, for terms depending on a small number of variables, or for certain cardinality-dependent terms.

A naive application of submodular minimization algorithms would not exploit the existence of specialized exchange capacity subroutines for individual terms. To overcome this, we cast the problem as a *submodular flow* (SF) problem in an auxiliary graph, and show that applying most existing SF algorithms would rely only on these subroutines.

We then explore in more detail Iwata's capacity scaling approach for submodular flows [18]. In particular, we show how to improve its complexity in the case when the function contains cardinality-dependent terms.

1 Introduction

In this paper we consider the problem of minimizing an objective function of the following form:

$$f(S) = \sum_{Q \in \mathcal{Q}^\circ} f_Q(S \cap Q) \quad \forall S \subseteq V \quad (1)$$

Here V is a set of nodes, $\mathcal{Q}^\circ \subseteq 2^V$ is a set of subsets of V , and $f_Q : 2^Q \rightarrow \mathbb{R}$ are submodular functions.

Function f is itself submodular, and thus can be minimized in polynomial time. The current fastest strongly polynomial algorithms are those of Orlin [24] and Iwata-Orlin [21], which take time $O(n^5 EO + n^6)$, where $n = |V|$ and EO is the time to run the value oracle for $f(S)$. The fastest weakly polynomial algorithms are those of Iwata [19] and Iwata-Orlin [21] which run in time $\tilde{O}(n^4 EO + n^5)$.

However, applying a general-purpose submodular minimization algorithm may not be the most efficient technique, since it does not exploit the special structure of f . It is often the case that terms f_Q have a special form that allow an efficient computation of exchange capacities, which are defined in the next section. Roughly speaking, this means that we can efficiently minimize function $f_Q(S) - z(S)$ for any vector $z \in \mathbb{R}^Q$. (As usual, $z(S)$ denotes $\sum_{i \in S} z_i$.) The main goal of this paper is to develop an algorithm that can exploit the existence of specialized exchange capacities subroutines.

To achieve this goal, we use the framework of *submodular flows* (SF) introduced by Edmonds and Giles [8]. We show that the problem of minimizing f can be cast as a particular SF instance in an auxiliary graph, so that computing exchange capacities for the new problem is equivalent to computing exchange capacities for individual terms f_Q . Most existing algorithms for submodular flows rely on the exchange capacity oracle, which gives the desired result.

We then present a capacity scaling technique for solving the problem. Its complexity is $O((n + \sum_Q \alpha_Q)(n + \sum_Q \beta_Q) \log U)$ where U is an upper bound on function values and α_Q, β_Q depend on the type of term f_Q :

- (a) If $|Q| = 2$ then $(\alpha_Q, \beta_Q) = (1, 1)$.
- (b) If $f_Q(S) = g(|S|)$ then $(\alpha_Q, \beta_Q) = (|Q|, |Q|)$. Note, $g(\cdot)$ must be concave.
- (c) If $f_Q(S) = g(|S \cap Q'|, |S \cap Q''|)$ where Q', Q'' are disjoint subsets of Q then $(\alpha_Q, \beta_Q) = (|Q|^2, |Q|)$.
- (d) For any other term f_Q we have $(\alpha_Q, \beta_Q) = (|Q|^2, |Q|^2 + |Q| \cdot h_Q)$ where h_Q is the time of the exchange capacity oracle for the (scaled version of) f_Q .

In (b) and (c) we assume that function g can be evaluated in $O(1)$ time. For cases (c) and (d) we use the scaling technique of Iwata [18].

Applications Functions with terms of the form (a)-(c) have recently appeared in computer vision applications. Terms (a) and (b) were used for the *image segmentation* problem [22, 25], while terms (a) and (c) were used for *co-segmenting* two images containing a similar object [17]. (The latter work used terms of the form $f_Q(S) = -c \cdot |S \cap Q'| \cdot |S \cap Q''|$ with $c \geq 0$.)

Note, objective functions used in computer vision very often have form (1) where $|Q|$ is quite small (2,3,...). Terms f_Q encode interactions between neighboring pixels. Currently, researchers restrict themselves to functions that can be reduced to a minimum s - t cut problem (see discussion in [26]), since minimizing general submodular functions is too expensive in practice. Our work may remove such restriction.

Related work The problem of minimizing functions of the form (1) was studied by Cooper [5], who formulated a linear program with an exponential number of constraints and showed that its optimal value coincides with the minimum of f . The formulation that we will use closely resembles that in [5]. Note, however, that the question of how to solve this LP efficiently was not addressed in [5], and a connection to the submodular flow problem was not given.

It is known that in certain cases the problem can be reduced to a minimum s - t cut problem in a graph with auxiliary nodes. Billionnet and Minoux [2] showed that this can be done for functions with cubic terms, i.e. when $|Q| \leq 3$ for all terms f_Q . Reductions for certain subclasses with higher order terms were given by Freedman and Drineas [10], Kohli et al. [22] and Živný and Jeavons [27]. The resulting maxflow problem could be solved e.g. in $O(\min(\hat{n}^{2/3}, \hat{m}^{1/2})\hat{m} \log(\hat{n}^2/\hat{m}) \log \hat{U})$ time by the algorithm of Goldberg and Rao [16], where \hat{n}, \hat{m} are the number of nodes and edges in the constructed graph and \hat{U} is a bound on edge capacities.

On the negative side, Živný et al. [26] proved that some submodular terms with $|Q| = 4$ do not admit such a reduction. Even if the reduction exists, it may result in a graph which would be prohibitively large in practice. Consider, for example, terms of the form $f_Q(S) = g(|S|)$ where g is concave. The reduction of Kohli et al. [22] adds b extra nodes and $b|Q|$ extra edges for each term f_Q , where b is the number of breakpoints of the piecewise-linear concave function g . If g is strictly concave (as in the application of [25]) then $b = |Q| - 1$, so there would be $O(|Q|^2)$ edges. In contrast, our technique uses only $O(|Q|)$ memory. The same holds for the function $f_Q(S) = -c \cdot |S \cap Q'| \cdot |S \cap Q''|$ used in [17].

Fujishige and Iwata [12] considered functions of the form $f(S) + g(|S|)$ on a distributive lattice where f is submodular and g is concave. It was shown that the problem is equivalent to a *parametric* problem: minimize function of form $f(S) + c_\lambda(S)$ for all values of λ , where $\{c_\lambda\}_\lambda$ is a certain family of non-increasing vectors in \mathbb{R}^V .

2 Problem formulation

Let \mathcal{Q} be the set obtained from \mathcal{Q}° by removing all singleton subsets of the form $\{i\}$, $i \in V$. Thus, $|Q| \geq 2$ for all $Q \in \mathcal{Q}$. Without loss of generality we assume that function f is given by

$$f(S) = \sum_{i \in S} c_{it} + \sum_{i \in V-S} c_{si} + \sum_{Q \in \mathcal{Q}} f_Q(S \cap Q) \quad (2)$$

where c_{it}, c_{si} are non-negative numbers and each term f_Q satisfies the following condition:¹

$$\min_{S \subseteq 2Q} f_Q(S) = f_Q(\emptyset) = f_Q(Q) = 0 \quad (3)$$

Base polyhedron and exchange capacities The base polyhedron [7] of f_Q is defined as

$$B(f_Q) = \{\varphi_Q \in \mathbb{R}^Q \mid \varphi_Q(S) \leq f_Q(S) \quad \forall S \subseteq Q, \quad \varphi_Q(Q) = f_Q(Q) = 0\} \quad (4)$$

Given a vector $\varphi_Q \in B(f_Q)$ and distinct nodes $i, j \in Q$, the exchange capacity \bar{c}_{Qij} is the maximum value of $\epsilon \geq 0$ such that the operation $\varphi_{Qi} := \varphi_{Qi} + \epsilon$, $\varphi_{Qj} := \varphi_{Qj} - \epsilon$ keeps φ_Q in $B(f_Q)$. Clearly,

$$\bar{c}_{Qij} = \min_{S \subseteq Q} \{\bar{f}_Q(S) \mid i \in S \subseteq Q - \{j\}\} \quad , \quad \bar{f}_Q(S) = f_Q(S) - \varphi_Q(S) \quad (5)$$

Computing \bar{c}_{Qij} is equivalent to minimizing a submodular function. This can be done in polynomial time by a number of general-purpose submodular minimization algorithms. Furthermore, for many choices of f_Q there exist more efficient specialized techniques.

A remark on notation: in this paper we always use “bar” (\bar{c}_{Qij} , \bar{f}_Q , ...) to indicate “residual” values, i.e. values that take into account current flow.

Maximum flow formulation Let us construct a directed capacitated graph $G = (N, A, c)$ as follows. The set of nodes will be $N = \{s, t\} \cup V \cup_{Q \in \mathcal{Q}} Q^*$ where s, t are the source and the sink and $Q^* = \{Qi \mid i \in Q\}$ is a unique copy of Q . Here Qi is a shorthand notation for the pair (Q, i) . The set of arcs will be

$$A = \{(i, Qi), (Qi, i) \mid i \in V, Qi \in N\} \cup \{(s, i), (i, s), (i, t), (t, i) \mid i \in V\}$$

Arc capacities c_{si}, c_{it} are the same as in (2). Arcs to the source and from the sink have zero capacity ($c_{is} = c_{ti} = 0$), and all “internal” arcs have infinite capacity ($c_{i, Qi} = c_{Qi, i} = +\infty$).

A flow φ is a vector in \mathbb{R}^A . For a subset $Q \in \mathcal{Q}$ we denote $\varphi_Q \in \mathbb{R}^Q$ to be the vector with components $\varphi_{Qi} = \varphi_{i, Qi}$. We also denote $value(\varphi) = \sum_{i \in V} \varphi_{si}$ to be the amount of flow sent from the source. We will consider the following maximum flow problem:

$$\max \quad value(\varphi) \quad \text{s.t.} \quad (6a)$$

$$\varphi_{uv} = -\varphi_{vu} \quad \forall (u, v) \in A \quad (\text{antisymmetry}) \quad (6b)$$

$$\varphi_a \leq c_a \quad \forall a \in A \quad (\text{capacity constraints}) \quad (6c)$$

$$\sum_{(u, i) \in A} \varphi_{ui} = 0 \quad \forall i \in V \quad (\text{flow conservation for } V) \quad (6d)$$

$$\varphi_Q \in B(f_Q) \quad \forall Q \in \mathcal{Q} \quad (\text{base polyhedron constraints}) \quad (6e)$$

Note, if φ is feasible then we also have $value(\varphi) = \sum_{i \in V} \varphi_{it}$ since $\sum_{i \in V} \varphi_{si} - \sum_{i \in V} \varphi_{it} = \sum_{i \in V} [\varphi_{si} + \varphi_{ti}] = -\sum_{i \in V} \sum_{Qi \in N} \varphi_{Qi, i} = \sum_{Q \in \mathcal{Q}} \sum_{i \in Q} \varphi_{i, Qi} = 0$.

¹ If term f_Q with $f_Q(\emptyset) = 0$ does not satisfy (3) then we can replace it with the sum $\varphi_Q(S) + \bar{f}(S)$ where $\bar{f}(S) = f(S) - \varphi_Q(S)$ and φ_Q is a vector in the base polyhedron of f_Q , which can easily be computed by a greedy algorithm of Edmonds [7].

The linear program (6) is very similar to that in [5], with some minor differences; for example, the “balance” constraint $\varphi_Q(Q) = 0$ is not present in [5].

The rest of the paper is organized as follows. Section 3 gives a reduction of problem (6) to a submodular flow problem, which leads to a number of algorithms for solving (6). Section 4 describes a pseudo-polynomial augmenting path algorithm, which is a specialization of the standard augmenting path algorithm for submodular flows. By analyzing the algorithm we will prove that the maximum of (6) coincides with the minimum of f . Section 5 presents a scaling version of the augmenting path algorithm, while section 6 discusses some implementational issues and states the complexity of the algorithm.

The reader may choose to skip the next section; familiarity with the submodular flow problem will not be necessary for understanding the augmenting path algorithm.

3 Reduction to a submodular flow problem

We will consider a directed capacitated graph $G' = (N, A', c)$ where $A' = A \cup \{(s, t), (t, s)\}$ and the capacities of the new arcs are $c_{ts} = +\infty$, $c_{st} = 0$. If $\varphi \in \mathbb{R}^{A'}$ is a flow in G' and u is a node in N then $\partial\varphi(u) = \sum_{(v,u) \in A'} \varphi_{vu}$ will denote the amount of flow that comes into u .

Let us recall a definition of a submodular flow problem for a graph G' [8, 15]. Assume that each arc $a \in A'$ has a cost d_a , and let $g : 2^N \rightarrow \mathbb{R}$ be a submodular function with $g(\emptyset) = g(N) = 0$. Then the problem is defined as

$$\max \quad \sum_{a \in A'} d_a \varphi_a \quad \text{s.t.} \quad (7a)$$

$$\varphi_{uv} = -\varphi_{vu} \quad \forall (u, v) \in A' \quad (7b)$$

$$\varphi_a \leq c_a \quad \forall a \in A' \quad (7c)$$

$$\partial\varphi \in B(g) \quad (7d)$$

where $B(g)$ is the base polyhedron of g :

$$B(g) = \{z \in \mathbb{R}^N \mid z(X) \leq g(X) \quad \forall X \subseteq N, \quad z(N) = 0\} \quad (8)$$

In order to simulate problem (6), we set arc costs as follows: $d_{ts} = 1$ and $d_a = 0$ for all other arcs a . Function g is defined by

$$g(X) = \sum_{Q \in \mathcal{Q}} f_Q(X^Q)$$

where we introduced notation $X^Q = \{i \in Q \mid Qi \in X\}$.

Proposition 1. *Problems (6) and (7) are equivalent.*

Proof. Suppose that $\varphi \in \mathbb{R}^A$ is a feasible flow for problem (6). Let us extend it to a flow in G' by setting $\varphi_{ts} = \text{value}(\varphi)$, $\varphi_{st} = -\text{value}(\varphi)$. Clearly, conditions (7b) and (7c) are satisfied. It is also easy to check that $z = \partial\varphi \in B(g)$. Indeed, we have $z_i = 0$ for $i \in V \cup \{s, t\}$ and $z_{Qi} = \varphi_{Qi}$ for $Qi \in N$. Conditions $\varphi_{Qi} \in B(f_Q)$ then imply that $z(N) = 0$ and for any $X \subseteq N$ there holds $z(X) = \sum_{Q \in \mathcal{Q}} \varphi_Q(X^Q) \leq \sum_{Q \in \mathcal{Q}} f_Q(X^Q) = g(X)$. Thus, φ is a feasible flow for problem (7). Furthermore, the values of objective functions of (6) and (7) coincide.

Conversely, suppose that $\varphi \in \mathbb{R}^{A'}$ is a feasible flow for problem (7); let us show that its restriction to A is feasible for problem (6). Conditions (6b) and (6c) follow from (7b) and (7c). Denote $z = \partial\varphi$. If X is a subset of N with $g(X) = g(N - X) = 0$ then $z \in B(g)$ implies $z(X) \leq g(X) = 0$ and $-z(X) = z(N - X) \leq g(N - X) = 0$, so $z(X) = 0$. Applying this fact for subset $X = \{i\}$ yields (6d), and applying this fact for subset $X = Q^*$ yields constraint $\varphi_Q(Q) = 0$, which is a part of (6e). Finally, if $S \subseteq Q$ then $\varphi_{Qi}(S) = z(S^*) \leq g(S^*) = f_Q(S)$ where we denoted $S^* = \{Qi \mid i \in S\}$. Thus, $\varphi_Q \in B(f_Q)$. \square

Exchange capacities Most submodular flow algorithms rely on the following operation: given a feasible flow $\varphi \in \mathbb{R}^{A'}$ with $z = \partial\varphi \in B(g)$ and distinct nodes $u, v \in N$, compute the exchange capacity $\bar{c}_{uv} = \min_X \{\bar{g}(X) \mid u \in X \subseteq N - \{v\}\}$ where $\bar{g}(X) = g(X) - z(X)$. The proposition below shows that computing these capacities is equivalent to computing exchange capacities \bar{c}_{Qij} for individual terms f_Q with respect to flow φ (given by eq. (5)).

Proposition 2. $\bar{c}_{uv} = \bar{c}_{Qij}$ if $(u, v) = (Qi, Qj)$ and $\bar{c}_{uv} = 0$ otherwise.

Proof. As shown above, $z_i = 0$ for $i \in V \cup \{s, t\}$, therefore $z(X) = \sum_{Q \in \mathcal{Q}} \varphi_{Qi}(X^Q)$ for all subsets $X \subseteq N$. This implies that

$$\bar{g}(X) = \sum_{Q \in \mathcal{Q}} \bar{f}_Q(X^Q) \quad (9)$$

The fact that $\varphi_Q \in B(f_Q)$ also implies $\min_{S \subseteq Q} \bar{f}_Q(S) = \bar{f}_Q(\emptyset) = \bar{f}_Q(Q) = 0$ for all $Q \in \mathcal{Q}$. Therefore, if $(u, v) = (Qi, Qj)$ then the minimization problem $\min_X \{\bar{g}(X) \mid u \in X \subseteq N - \{v\}\}$ has a minimizer $X \subseteq Q^*$, and thus $\bar{c}_{uv} = \min_X \{\bar{g}(X) \mid u \in X \subseteq Q^* - \{v\}\} = \bar{c}_{Qij}$. Now suppose that $(u, v) \neq (Qi, Qj)$. Let $U \subset N$ be the “completion” of u : $U = \{u\}$ if $u \in V \cup \{s, t\}$ and $U = Q^*$ if $u = Qi$. There holds $v \notin U$ since we assumed that $(u, v) \neq (Qi, Qj)$ and u, v are distinct. We have $\bar{g}(U) = 0$, and thus $\bar{c}_{uv} = 0$. \square

Problem (7) is actually a *maximum submodular flow* problem, which is a special case of the more general *minimum cost submodular flow* problem (see survey [15]). The former problem can be solved in time $O(|N|^3 h)$ by a push-relabel method of Fujishige and Zhang [13], where h is the time of the exchange capacity oracle (see also [20], section 3.1). Clearly, for certain functions f this complexity can be better than bounds $O(n^5 EO + n^6)$ and $\tilde{O}(n^4 EO + n^5)$ for submodular function minimization.

In our case h is the maximum time of oracles over individual terms. This appears to be a rather crude way of estimating the complexity, as it does not take into account the structure of individual terms. We conjecture that a more careful analysis of the algorithm can give a bound which better illustrates contributions of individual terms. In the subsequent sections we will give an example of such a bound for a capacity scaling augmenting path algorithm applied to problem (6).

4 Augmenting path algorithm

A shortest augmenting path algorithm for a problem equivalent to maximum submodular flows was given by Fujishige [14]. We now describe its application to problem (6), and prove that the value of the maximum flow coincides with the minimum of f . We will generalize the problem slightly: we assume that capacities c_{is} and c_{ti} are non-negative numbers which are not necessarily zero. (We will need this in the next section.)

Given a flow φ , the residual capacity for arc $a \in A$ is defined as $\bar{c}_a = c_a - \varphi_a$. Similarly, we define “residual functions” \bar{f}_Q by $\bar{f}_Q(S) = f_Q(S) - \varphi_Q(S)$ for $S \subseteq Q$. It can be seen that if φ satisfies antisymmetry and conservation constraints (6b), (6d) then for any $S \subseteq V$ there holds

$$f(S) = \text{value}(\varphi) + \sum_{i \in S} \bar{c}_{it} + \sum_{i \in V-S} \bar{c}_{si} + \sum_{Q \in \mathcal{Q}} \bar{f}_Q(S \cap Q) \quad (10)$$

Indeed, subtracting (2) from (10) gives $\sum_{i \in V} \varphi_{si} - \sum_{i \in S} \varphi_{it} - \sum_{i \in V-S} \varphi_{si} - \sum_{Q \in \mathcal{Q}} \sum_{i \in S \cap Q} \varphi_{i, Qi} = \sum_{i \in S} [\varphi_{si} + \varphi_{ti} + \sum_{Q \in \mathcal{Q}} \varphi_{Qi, i}] = 0$. All residual values for a feasible φ are non-negative, so equation (10) implies the weak duality relationship:

$$\max\{\text{value}(\varphi) \mid \varphi \text{ is feasible}\} \leq \min\{f(S) \mid S \subseteq V\} \quad (11)$$

Given a feasible flow φ , let \bar{A} be the following set of arcs:

$$\bar{A} = \{a \in A \mid \bar{c}_a > 0\} \bigcup_{Q \in \mathcal{Q}} \bar{A}_Q, \quad \bar{A}_Q = \{(Qi, Qj) \mid i, j \in Q, i \neq j, \bar{c}_{Qij} > 0\} \quad (12)$$

Proposition 3. *If there is no path from s to t in (N, \bar{A}) then the set $S = \{i \in V \mid i \text{ is reachable from } s \text{ in } (N, \bar{A})\}$ satisfies $f(S) = \text{value}(\varphi)$, and therefore φ is a maximum flow and S is a minimizer of f .*

Proof. It suffices to show that every term in the RHS of (10) (except maybe for the first term $\text{value}(\varphi)$) is zero. If $i \in S$ then $\bar{c}_{it} = 0$, otherwise t would be reachable from s . If $i \in V - S$ then $\bar{c}_{si} = 0$, otherwise i would belong to S . Consider the term for subset $Q \in \mathcal{Q}$, and denote $S' = S \cap Q$. For each pair of nodes $i \in S'$, $j \in Q - S'$ function \bar{f}_Q must have a minimizer S_{ij} with $i \in S_{ij} \subseteq Q - \{j\}$, otherwise we would have $\bar{c}_{Qij} > 0$ so node j could be reached from i via arcs $(i, Qi), (Qi, Qj), (Qj, j) \in \bar{A}$ and thus j would be in S . The submodularity of \bar{f}_Q implies that the set $\bigcup_{i \in S'} \bigcap_{j \in Q - S'} S_{ij}$ is a minimizer of \bar{f}_Q as well. The latter set coincides with $S' = S \cap Q$, therefore $\bar{f}_Q(S \cap Q) = 0$. \square

Now suppose that there exists a path P from s to t ; such a path is called an *augmenting path*. Clearly, we can send some flow $\delta > 0$ along the path² so that the flow would remain feasible and $\text{value}(\varphi)$ would increase by δ . This leads to

Proposition 4 (Strong duality). *The value of the maximum flow in (6) coincides with the minimum of f .*

Proof. Let φ be a maximum flow. There can be no augmenting path for φ , otherwise φ would not be maximal. The claim now follows from proposition 3. \square

From now on, we assume that all capacities c_{si} , c_{it} and values $f_Q(S)$ for $S \subseteq Q$ are integers bounded by constant U . A maximum flow can then be computed in pseudo-polynomial time by the following augmenting path algorithm:

- S0 Set $\varphi_a = 0$ for all arcs a .
- S1 Construct set of arcs \bar{A} as in (12).
- S2 Find a shortest path P from s to t in (N, \bar{A}) ; if no such P exists, terminate.
- S3 Send 1 unit of flow along P and go to step 1.

Note, it is well-known that for integer-valued submodular flow problems sending 1 unit of flow along a **shortest** augmenting path preserves flow feasibility [14]. In our case we can relax slightly the requirement that P is shortest; we only need P to be *minimal*:

Definition 5. *Let P be a simple (i.e. node-disjoint) path in (N, \bar{A}) . We call P minimal (with respect to (N, \bar{A})) if the following property holds: if $(Qi, Qj), (Qi', Qj')$ are two distinct arcs in the path (occurring in this order) then \bar{A} does not have arc (Qi, Qj') .*

Clearly, any shortest augmenting path from s to t is minimal. In Appendix A we prove that sending one unit of flow from s to t along a minimal path preserves flow feasibility.

It is not difficult to show that sets \bar{A}_Q are transitive, i.e. $(i, j), (j, k) \in \bar{A}_Q$ implies $(i, k) \in \bar{A}_Q$ (see Appendix A). Thus, if P is minimal then $(Qi, Qj) \in P$ implies that the previous arc in P is (i, Qi) and the next arc is (Qj, j) . The operation of sending flow through these three arcs will be referred to as “sending flow from i to j via Q ”.

² Sending flow δ along arc $(u, v) \in A$ denotes the operation $\varphi_{uv} := \varphi_{uv} + \delta$, $\varphi_{vu} := \varphi_{vu} - \delta$. Sending flow δ along arc $(Qi, Qj) \in \bar{A}_Q$ does not change φ .

S0 For each $Q \in \mathcal{Q}$ set $\varphi_Q := \text{AdjustFlow}_Q^\Delta(\varphi_Q)$ to make sure that $\varphi_Q \in B(f_Q^\Delta)$. Adjust other flow components so that φ satisfies antisymmetry and flow conservation constraints:

- Set $\varphi_{Qi,i} := -\varphi_{i,Qi}$ for all $Qi \in N$.
- For each node $i \in V$ compute $\delta = \sum_{(u,i) \in A} \varphi_{ui}$; if $\delta > 0$, send δ units of flow back to the source via arc (i, s) , otherwise send $-\delta$ units of flow from the sink via arc (t, i) .

S1 Construct set of arcs \bar{A}^Δ as follows:

$$\bar{A}^\Delta = \{(u, v) \in A \mid \bar{c}_{uv} \geq \lceil \Delta \rceil\} \bigcup_{Q \in \mathcal{Q}} \bar{A}_Q^\Delta \quad (13)$$

S2 Find minimal path P in (N, \bar{A}^Δ) ; if no such path exists, terminate.

S3 Send $\lceil \Delta \rceil$ units of flow along P and go to step S1.

Figure 1: **Δ -phase.** Definitions of function f_Q^Δ , procedure $\text{AdjustFlow}_Q^\Delta(\varphi)$ and set \bar{A}^Δ for different types of terms f_Q are given in sections 5.1-5.3.

5 Capacity scaling algorithm

We now apply a scaling technique to get a weakly-polynomial algorithm. As usual, the algorithm works in phases. Each phase is associated with a number $\Delta = 2^l$, $l = -1, 0, 1, 2, \dots$; we call it a Δ -phase. To initialize, we set $\Delta = 2^{\lceil \log_2 U \rceil}$ and $\varphi_a = 0$ for all arcs $a \in A$. After completing the Δ -phase we divide Δ by 2 and proceed to the next phase (or terminate, if $\Delta = 1/2$). The Δ -phase is described in Figure 1. This description uses the following yet undefined objects:

- f_Q^Δ is a submodular function. When $\Delta = \frac{1}{2}$, function f_Q^Δ coincides with f_Q .
- $\text{AdjustFlow}_Q^\Delta(\varphi_Q)$ is a procedure that outputs a vector in $B(f_Q^\Delta)$ whose components are integer multiples of $\lceil \Delta \rceil$.
- \bar{A}_Q^Δ is a subset of arcs of the form (Qi, Qj) where i, j are distinct nodes in Q . Set \bar{A}_Q^Δ is transitive, i.e. $(Qi, Qj), (Qj, Qk) \in \bar{A}_Q^\Delta$ for distinct $i, j, k \in Q$ implies $(Qi, Qk) \in \bar{A}_Q^\Delta$. When $\Delta = \frac{1}{2}$, set \bar{A}_Q^Δ coincides with the set \bar{A}_Q defined in (12).

Definitions of these three objects will depend on the type of term f_Q ; different cases are considered in sections 5.1-5.3. Set \bar{A}_Q^Δ will be defined in such a way that each augmentation keeps flow φ_Q in $B(f_Q^\Delta)$.

It is clear that each Δ -phase maintains the following invariants: (i) components of flow φ are integer multiples of $\lceil \Delta \rceil$; (ii) φ is a feasible Δ -flow, i.e. it satisfies antisymmetry (6b), capacity (6c), flow conservation constraints (6d), as well as base polyhedron constraints $\varphi_Q \in B(f_Q^\Delta)$. (We assume that capacities c_{is}, c_{ti} are infinite, so that sending flow to the source or from the sink in step S0 is always feasible). To estimate the complexity, we will use values α_Q (to be defined in sections 5.1-5.3) that satisfy

$$\bar{f}_Q^\Delta(S) + \sum_{i \in Q} |\varphi_{Qi} - \varphi_{Qi}^\circ| \leq \alpha_Q \cdot \lceil \Delta \rceil \quad (14)$$

where φ° is the flow in the beginning of Δ -phase, S is the set of nodes in Q reachable from s in the graph $(N, \bar{A}^{2\Delta})$ constructed with respect to flow φ° , $\varphi = \text{AdjustFlow}_Q^\Delta(\varphi^\circ)$ and $\bar{f}_Q^\Delta(S) =$

$f_Q^\Delta(S) - \varphi_Q(S)$. Values α_Q can be used for estimating the number of augmentations (a proof is given in Appendix B):

Proposition 6. *Each Δ -phase terminates after at most $2n + \sum_{Q \in \mathcal{Q}} \alpha_Q$ augmentations, and so the whole algorithm performs $O((2n + \sum_{Q \in \mathcal{Q}} \alpha_Q) \log U)$ augmentations.*

To complete the description of the algorithm, we need to provide constructions for different types of terms f_Q . In sections 5.1-5.3 below we consider three types: pairwise terms, cardinality-dependent terms and general terms.

5.1 Pairwise terms

First, we consider the case when $|Q| = 2$, which occurs very frequently in applications (see e.g. [3] for a survey of applications in computer vision). We define $f_Q^\Delta = f_Q$ for all Δ . This means that procedure $\text{AdjustFlow}_Q^\Delta(\varphi_Q)$ can simply return φ_Q - it is guaranteed to belong to $B(f_Q^\Delta) = B(f_Q^{2\Delta})$. Let $Q = \{i, j\}$. Constraint $\varphi_Q \in B(f_Q^\Delta)$ can be written as

$$\varphi_{Qi} \leq f_Q(\{i\}) \quad \varphi_{Qj} \leq f_Q(\{j\}) \quad \varphi_{Qi} + \varphi_{Qj} = 0 \quad (15)$$

The set of arcs \bar{A}_Q^Δ is constructed as follows: we add arc (i, j) if $\bar{c}_{Qij} = \bar{f}_Q(\{i\}) \geq \lceil \Delta \rceil$, and arc (j, i) if $\bar{c}_{Qji} = \bar{f}_Q(\{j\}) \geq \lceil \Delta \rceil$. Clearly, we can always push $\lceil \Delta \rceil$ units of flow through the added arcs - constraints (15) will be preserved.

It is easy to see that we can take $\alpha_Q = 2$. Indeed, let S be the set used in eq. (14). Since $\varphi = \varphi^\circ$, we need to show that $\bar{f}_Q(S) \leq 2\lceil \Delta \rceil$. If $S = \{i\}$ then $(Qi, Qj) \notin \bar{A}_Q^{2\Delta}$, therefore $\bar{f}_Q(S) \leq \lceil 2\Delta \rceil - 1 \leq 2\lceil \Delta \rceil$. The case $S = \{j\}$ is similar. If S is empty or equals Q then $\bar{f}_Q(S) = 0$.

5.2 Cardinality-dependent terms

Let us now assume that $f_Q(S)$ for $S \subseteq Q$ depends only on $|S|$. Thus, $f_Q(S) = g(|S|)$ where g is a concave function. As above, we define $f_Q^\Delta = f_Q$ for all Δ , and accordingly procedure $\text{AdjustFlow}(\varphi_Q)$ simply returns φ_Q . Below we describe how to construct set \bar{A}_Q^Δ .

For integer numbers $a \leq b$ let $[a..b]$ be the set of integers in $[a, b]$. We can assume that $g(\cdot)$ is defined only on $[0..m]$ where $m = |Q|$. We denote $z = \varphi_Q$, so $z_i = \varphi_{i, Qi} = -\varphi_{Qi, i}$ for $i \in Q$. For a vector $z \in \mathbb{R}^Q$ we also denote (z^1, \dots, z^m) to be the sequence of values of z_i sorted in the non-increasing order. Thus, z^k is the k -th largest number among values z_i , $i \in Q$. For a node $i \in Q$ define

$$L(i) = \min\{k \in [1..m] \mid z^k = z_i\}, \quad R(i) = \max\{k \in [1..m] \mid z^k = z_i\}$$

Let us define “residual” function $\bar{g}(\cdot)$ by $\bar{g}(k) = \min\{\bar{f}_Q(S) \mid S \subseteq Q, |S| = k\}$ for $k \in [0..m]$. We have

$$\bar{g}(k) = g(k) - \max\{z(S) \mid S \subseteq Q, |S| = k\} = g(k) - \sum_{k'=1}^k z^{k'} \quad (16)$$

Clearly, constraint $z \in B(f_Q)$ is equivalent to the following conditions: (i) function $\bar{g}(\cdot)$ is non-negative, i.e. $\bar{g}(k) \geq 0$ for all $k \in [0..m]$; (ii) $z(Q) = 0$.

Recall that sending flow $\lceil \Delta \rceil$ from i to j via Q denotes the following operation: $z_i := z_i + \lceil \Delta \rceil$, $z_j := z_j - \lceil \Delta \rceil$. Next, we describe the effect of this operation on function $\bar{g}(\cdot)$. Three cases are possible (we assume that we are in a Δ -phase, so all components of vector z are integer multiples of $\lceil \Delta \rceil$):

- $z_i \leq z_j - 2\lceil \Delta \rceil$. The change in the sequence (z^1, \dots, z^m) is

$$(\dots, 0, -\lceil \Delta \rceil, 0, \dots, 0, +\lceil \Delta \rceil, 0, \dots)$$

where $-\lceil \Delta \rceil$ is in the position $R(j)$ and $+\lceil \Delta \rceil$ is in the position $L(i)$. Therefore, the effect of the operation is that all values $\bar{g}(k)$ for $k \in [R(j)..L(i) - 1]$ are increased by $\lceil \Delta \rceil$.

- $z_i = z_j - \lceil \Delta \rceil$. The values z_i and z_j are swapped, therefore the sequence (z^1, \dots, z^m) and function $\bar{g}(\cdot)$ do not change.
- $z_i \geq z_j$. The change in the sequence (z^1, \dots, z^m) is

$$(\dots, 0, +\lceil \Delta \rceil, 0, \dots, 0, -\lceil \Delta \rceil, 0, \dots)$$

where $+\lceil \Delta \rceil$ is in the position $L(i)$ and $-\lceil \Delta \rceil$ is in the position $R(j)$. Therefore, all values $\bar{g}(k)$ for $k \in [L(i)..R(j) - 1]$ are decreased by $\lceil \Delta \rceil$.

In the first two cases function $\bar{g}(\cdot)$ cannot become negative, thus sending $\lceil \Delta \rceil$ units of flow from i to j via Q is always possible if $z_i < z_j$. Accordingly, we add arcs (i, j) to \bar{A}_Q^Δ for all pairs of nodes $i, j \in Q$ with $z_i < z_j$. If $z_i \geq z_j$ then we can send flow if and only if $\min_{k \in [L(i)..R(j)-1]} \bar{g}(k) \geq \lceil \Delta \rceil$. However, if we add all arcs that satisfy this constraint then sending $\lceil \Delta \rceil$ units of flow through **multiple** arcs of Q along a minimal path could make some values $\bar{g}(k)$ negative. To prevent this, we add to \bar{A}_Q^Δ those arcs (i, j) with $z_i \geq z_j$ that satisfy the following constraint:

$$\min_{k \in [L(i)..R(j)-1]} \bar{g}(k) \geq 3\Delta/2 \quad (17)$$

The proposition below shows the correctness of this construction, and gives a bound on α_Q . A proof is given in Appendix C.

Proposition 7. (a) Set \bar{A}_Q^Δ is transitive. (b) Sending $\lceil \Delta \rceil$ units of flow through a minimal path P in (N, \bar{A}^Δ) preserves constraint $\varphi_Q = z \in B(f_Q)$. (c) Eq. (14) is satisfied by $\alpha_Q = 3(m-1)$.

5.3 General submodular terms

For general terms we can use the technique of Iwata [18]. f_Q^Δ is defined as

$$f_Q^\Delta(S) = \Delta \cdot \lfloor f_Q(S)/\Delta \rfloor + \lfloor \Delta \rfloor \cdot b(S) \quad \forall S \subseteq Q \quad (18)$$

where $b(S) = |S| \cdot |Q - S|$. As shown in [18], this function is submodular. The set \bar{A}_Q^Δ includes all arcs (Qi, Qj) that have non-zero residual capacity with respect to function $\bar{f}_Q^\Delta(S) = f_Q^\Delta(S) - \varphi_Q(S)$. Clearly, values of $f_Q^\Delta(S)$ are integer multiples of $\lceil \Delta \rceil$, so results in section 4 imply that pushing $\lceil \Delta \rceil$ of flow through a minimal path in (N, \bar{A}^Δ) preserves constraint $\varphi_Q \in B(f_Q^\Delta)$.

Procedure **AdjustFlow** $^\Delta(\varphi_Q^\circ)$ works as follows. First, define vector φ'_Q by $\varphi'_{Qi} = \varphi_{Qi}^\circ - m\lceil \Delta \rceil$ where $m = |Q|$. Vector φ'_Q belongs to submodular polyhedron

$$P(f_Q^\Delta) = \{\varphi_Q \in \mathbb{R}^Q \mid \varphi_Q(S) \leq f_Q^\Delta(S) \quad \forall S \subseteq Q\} \quad (19)$$

Indeed, for any $S \subseteq Q$ we have $\varphi'_Q(S) = \varphi_Q^\circ(S) - m\lceil \Delta \rceil \cdot |S| \leq f_Q^{2\Delta}(S) - m\lceil \Delta \rceil \cdot |S| \leq f_Q^\Delta(S)$. Since $\varphi'_Q \in P(f_Q^\Delta)$, there exists vector $\varphi_Q \in B(f_Q^\Delta)$ with $\varphi'_Q \geq \varphi_Q$, which can be found by a greedy algorithm starting from φ'_Q [11, Theorem 3.19]. This φ_Q is taken as the output of **AdjustFlow** $^\Delta(\varphi_Q^\circ)$.

It can be seen that $\alpha_Q = O(m^2)$. This follows from three facts: (1) $\bar{f}_Q^{2\Delta}(S) = 0$ where S is the set used in eq. (14) and $\bar{f}_Q^{2\Delta}$ is the residual function with respect to flow φ° ; (2) $|f_Q^{2\Delta}(S) - f_Q^\Delta(S)| = O(m^2\lceil \Delta \rceil)$; (3) $\sum_{i \in Q} |\varphi_i - \varphi_i^\circ| \leq m^2\lceil \Delta \rceil + \sum_{i \in Q} |\varphi_{Qi} - \varphi'_{Qi}| = m^2\lceil \Delta \rceil + \varphi_Q(Q) - \varphi'_Q(Q) = 2m^2\lceil \Delta \rceil$.

Note, procedure **AdjustFlow** $^\Delta(\varphi_Q)$ used in [18] is slightly more complicated; in particular it takes into account set S used in (14). However, both techniques lead to $\alpha_Q = O(m^2)$.

6 Efficient implementation

We now discuss how implement steps S1 and S2 of the algorithm, i.e. how to find a minimal augmenting path. Set \bar{A}^Δ contains $O(n + \sum_Q |Q|^2)$ arcs, so a naive computation would take $O(n + \sum_Q |Q|^2)$ time. However, this can easily be improved: it can be seen that an explicit construction of \bar{A}^Δ is not required.

We will use a breadth-first search (BFS) for computing a shortest path from s to t in (N, \bar{A}^Δ) . Each node $Qi \in N$ will have flag $\text{REACHED}(Qi)$, which is set to **false** at the beginning of BFS. We assume that each term f_Q supports operation $\text{GetNeighbors}_Q^\Delta(Qi)$ for a node $Qi \in N$ with $\text{REACHED}(Qi) = \text{false}$. This operation is defined as follows:

- Compute $S = \{Qj \in N \mid (Qi, Qj) \in \bar{A}_Q^\Delta, \text{REACHED}(Qj) = \text{false}\}$.
- Set $\text{REACHED}(Qj) := \text{true}$ for $Qj \in S \cup \{Qi\}$.
- Return S as a linked list.

Flags $\text{REACHED}(Qi)$ will not be modified by any other operation (except that they are reset to **false** at the beginning of BFS).

It is straightforward to implement the BFS procedure using operations $\text{GetNeighbors}_Q^\Delta(Qi)$. The running time of one augmentation (steps S1-S3) will then be $O(n + \sum_{Q \in \mathcal{Q}} \beta_Q)$ where β_Q for a fixed $Q \in \mathcal{Q}$ is the combined time taken by calls to $\text{GetNeighbors}_Q^\Delta(Qi)$, plus the time for sending flow through Q in step S3 (which may update internal structures for Q). In Appendix D we show how to implement $\text{GetNeighbors}_Q^\Delta(Qi)$ so that $\beta_Q = O(|Q|)$ in the following cases:

- $f_Q(S) = g(|S|)$ for $S \subseteq Q$.
- $f_Q(S) = g(|S \cap Q'|, |S \cap Q''|)$ where Q', Q'' are disjoint subsets of Q .

The second case relies on the algorithm of Aggarwal et al. [1] which computes row minima of a totally monotone matrix in linear time. For a general submodular term f_Q a naive implementation of $\text{GetNeighbors}_Q^\Delta(Qi)$ would make $|Q| - 1$ calls to the exchange capacity oracle for f_Q^Δ , giving $\beta_Q = O(|Q|^2 h_Q)$ where h_Q is the oracle's complexity. However, the set $\{(Qi, Qj) \mid (Qi, Qj) \in \bar{A}_Q^\Delta\}$ can alternatively be obtained from the minimal minimizer in $\arg \min \{\bar{f}_Q^\Delta(S) \mid i \in S \subseteq Q\}$. It is natural to assume that computing such minimal minimizer also takes time h_Q . Under this assumption $\beta_Q = O(|Q|^2 + |Q| \cdot h_Q)$. Combined with proposition 6, this leads to the overall complexity stated in the introduction.

7 Conclusions and future work

In recent years there has been an increased interest in the computer vision community in using submodular functions of the form (1) with high-order terms [22, 23, 25, 17, 6]. So far, researchers restricted themselves to functionals that can be transformed to pairwise terms by introducing auxiliary variables. The main goal of this paper is to advocate a more direct approach which could extend the set of practically tractable functionals.

To our knowledge, our bound $O((n + \sum_Q \alpha_Q)(n + \sum_Q \beta_Q) \log U)$ is the first one for minimization problem (1) that shows contributions of individual terms. It is quite likely, however, that it can be improved further. Indeed, the capacity scaling algorithm of Iwata [18] that we built on is not a state-of-the-art. In the future we plan to investigate applications of alternative submodular flow algorithms, such as the capacity scaling algorithm of Fleischer et al. [9] that improves on [18], or the push-relabel method of Fujishige and Zhang [13].

Appendix A: Minimal augmenting paths

First, let us show the set \bar{A}_Q defined in (12) is transitive, i.e. if i, j, k are distinct nodes in Q then $(Qi, Qj), (Qj, Qk) \in \bar{A}_Q$ implies $(Qi, Qk) \in \bar{A}_Q$. Suppose not, then $\bar{c}_{Qik} = 0$. This means that $\bar{f}_Q(S) = 0$ for some subset S with $i \in S \subseteq Q - \{k\}$. If $j \in S$ then $\bar{c}_{Qjk} = 0$ and $(Qj, Qk) \notin \bar{A}_Q$, and if $j \notin S$ then $\bar{c}_{Qij} = 0$ and $(Qi, Qj) \notin \bar{A}_Q$ - a contradiction.

Assume that the problem is integer-valued. It is straightforward to check that sending one unit of flow along a minimal path in (N, \bar{A}) from s to t preserves antisymmetry (6b), capacity (6c) and flow conservation (6d) constraints. We now prove that if P is a minimal path in (N, \bar{A}) whose endpoints belong to V then sending one unit of flow along P preserves base polyhedron constraints (6e). Note, P is *not* an augmenting path: it does not go from s to t . However, the operation of sending flow along P and the minimality of P are still well-defined.

We use induction on the length of P . If P is empty then the claim is trivial. Suppose P is not empty; since P is minimal and \bar{A}_Q are transitive, P must have the form $P = P_1 P_2$ where $P_1 = ((i, Qi), (Qi, Qj), (Qj, j))$ and i, j are distinct nodes in $Q \in \mathcal{Q}$. Since $(Qi, Qj) \in \bar{A}_Q$, sending one unit of flow along P_1 preserves base polyhedron constraints. We prove below that after sending this flow P_2 remains a minimal path in (N, \bar{A}) ; the claim will then follow by the induction hypothesis.

Clearly, we need to consider only arcs in \bar{A}_Q - subsets $\bar{A}_{Q'}$ for $Q' \in \mathcal{Q} - \{Q\}$ are not affected. Let us denote \hat{f}_Q to be the residual function after sending the flow and \hat{A}_Q to be the corresponding set of arcs. We have $\hat{f}_Q(S) = f_Q(S) - [i \in S] + [j \notin S]$ for $S \subseteq Q$, where $[\cdot]$ is the Iverson bracket: it is 1 if its argument is true, and 0 otherwise. We need to show two facts:

- (a) if $(Qk, Ql) \in P_2$ then (Qk, Ql) remains in \hat{A}_Q ;
- (b) if (Qk, Ql) and (Qk', Ql') are two distinct arcs in P_2 occuring in this order then arc (Qk, Ql') still does not belong to \hat{A}_Q .

Proof of claim (a) For a set $S \subseteq Q$ denote $[S] = ([i \in S], [j \in S], [k \in S], [l \in S])$. If the claim is false then there exists S with $[S] = (?, ?, 1, 0)$ and $\hat{f}_Q(S) = 0$. Since $(Qk, Ql) \in \bar{A}_Q$ before sending the flow, we must have $\bar{f}_Q(S) = \hat{f}_Q(S) + [i \in S] - [j \notin S] > 0$, therefore $[S] = (1, 0, 1, 0)$ and $\bar{f}_Q(S) = 1$. By minimality of P arc (Qi, Ql) was not in \bar{A}_Q before sending the flow, therefore there exists another set S' with $[S'] = (1, ?, ?, 0)$ and $\bar{f}_Q(S') = 0$. Since $(Qi, Qj), (Qk, Ql) \in \bar{A}_Q$ we must have $[S'] = (1, 1, 0, 0)$.

By submodularity $\bar{f}_Q(S \cap S') + \bar{f}_Q(S \cup S') \leq \bar{f}_Q(S) + \bar{f}_Q(S') = 1$, so one of the set $S \cap S'$, $S \cup S'$ is a minimizer of \bar{f} . We have $[S \cap S'] = (1, 0, 0, 0)$ and $[S \cup S'] = (1, 1, 1, 0)$, so either $(Qi, Qj) \notin \bar{A}_Q$ or $(Qk, Ql) \notin \bar{A}_Q$ - a contradiction.

Proof of claim (b) For a set $S \subseteq Q$ denote $[S] = ([i \in S], [j \in S], [k \in S], [l \in S], [k' \in S], [l' \in S])$. Arcs (Qi, Ql') and (Qk, Ql') are not in \bar{A}_Q before sending the flow, therefore there exist sets S and S' with $[S] = (1, ?, ?, ?, 0, 0)$, $[S'] = (?, ?, 1, ?, ?, 0)$ and $\bar{f}_Q(S) = \bar{f}_Q(S') = 0$. We have $(Qi, Qj), (Qk, Ql), (Qk', Ql') \in \bar{A}_Q$, therefore $[S] = (1, 1, ?, ?, 0, 0)$, $[S'] = (?, ?, 1, 1, 0, 0)$.

Consider set $S'' = S \cup S'$ with $[S''] = (1, 1, 1, 1, 0, 0)$. Sets S and S' are minimizers of a submodular function \bar{f} , and thus so is S'' . We have $\hat{f}_Q(S'') = \bar{f}_Q(S'') - [i \in S''] + [j \notin S''] = 0 - 1 + 1 = 0$, which implies the claim.

Appendix B: proof of proposition 6

Let φ° be the input flow to the Δ -phase, S to be the set of nodes in V reachable from s in $(N, \bar{A}^{2\Delta})$ and $\varphi = \text{AdjustFlow}(\varphi^\circ)$. Let $\bar{c}_{si}, \bar{c}_{it}$ and \bar{f}_Q^Δ be residual capacities and functions with respect to flow φ , and $\bar{c}_{si}^\circ, \bar{c}_{it}^\circ$ be residual capacities with respect to flow φ° . When the previous 2Δ -phase terminated, there were no augmenting paths from s to t in $(N, \bar{A}^{2\Delta})$, hence $\bar{A}^{2\Delta}$ cannot have arcs

(i, t) for $i \in S$ and (s, i) for $i \in V - S$. Therefore, $\bar{c}_{ii}^\circ \leq \lceil 2\Delta \rceil - 1$ for $i \in S$ and $\bar{c}_{is}^\circ \leq \lceil 2\Delta \rceil - 1$ for $i \in V - S$. Define

$$f^\Delta(S) = \text{value}(\varphi) + \sum_{i \in S} \bar{c}_{it} + \sum_{i \in V-S} \bar{c}_{si} + \sum_{Q \in \mathcal{Q}} \bar{f}_Q^\Delta(S \cap Q) \quad (20)$$

Each augmentation in the Δ -phase preserves this equality (assuming that $\bar{c}_{si}, \bar{c}_{it}$ and \bar{f}_Q^Δ are updated accordingly). All residual values stay non-negative, therefore the number of augmentations cannot exceed $(f^\Delta(S) - \text{value}(\varphi)) / \lceil \Delta \rceil$. Using (20) and the definition of step S0, we can write

$$\begin{aligned} f^\Delta(S) - \text{value}(\varphi) &\leq \sum_{i \in S} \bar{c}_{it}^\circ + \sum_{i \in V-S} \bar{c}_{si}^\circ + \sum_{Q \in \mathcal{Q}} \left[\bar{f}_Q^\Delta(S \cap Q) + \sum_{i \in Q} |\varphi_{Qi} - \varphi_{Qi}^\circ| \right] \\ &\leq n \cdot (\lceil 2\Delta \rceil - 1) + \sum_{Q \in \mathcal{Q}} \alpha_Q \cdot \lceil \Delta \rceil \leq \left(2n + \sum_{Q \in \mathcal{Q}} \alpha_Q \right) \cdot \lceil \Delta \rceil \end{aligned}$$

Appendix C: Proof of proposition 7

Proof of part (a) Let i, i', i'' be distinct nodes in Q and $(Qi, Qi'), (Qi', Qi'') \in \bar{A}_Q^\Delta$. If $z_i < z_{i''}$ then obviously $(Qi, Qi'') \in \bar{A}_Q^\Delta$. Suppose $z_i \geq z_{i''}$; in order to show $(Qi, Qi'') \in \bar{A}_Q^\Delta$, we need to prove that $\min_{k \in [L(i)..R(i'')-1]} \bar{g}(k) \geq 3\Delta/2$. Value $z_{i'}$ falls in one of the three intervals $[z_i, +\infty), (-\infty, z_{i'']}, (z_{i''}, z_i)$. These three cases are considered below.

- $z_{i'} \geq z_i \geq z_{i''}$. Since arc (Qi', Qi'') belongs to \bar{A}_Q^Δ and $z_{i'} \geq z_{i''}$, we must have

$$\min_{k \in [L(i')..R(i'')-1]} \bar{g}(k) \geq 3\Delta/2$$

The claim then follows from the fact that $L(i) \geq L(i')$ and so $[L(i)..R(i'')-1] \subseteq [L(i')..R(i'')-1]$.

- $z_i \geq z_{i''} \geq z_{i'}$. Since arc (Qi, Qi') belongs to \bar{A}_Q^Δ and $z_i \geq z_{i'}$, we must have

$$\min_{k \in [L(i)..R(i')-1]} \bar{g}(k) \geq 3\Delta/2$$

The claim then follows from the fact that $R(i') \geq R(i'')$ and so $[L(i)..R(i'')-1] \subseteq [L(i)..R(i')-1]$.

- $z_i > z_{i'} > z_{i''}$. We must have

$$\min_{k \in [L(i)..R(i')-1]} \bar{g}(k) \geq 3\Delta/2 \quad \min_{k \in [L(i')..R(i'')-1]} \bar{g}(k) \geq 3\Delta/2$$

The claim then follows from the fact that $R(i') \geq L(i')$ so $[L(i)..R(i')-1] \cup [L(i')..R(i'')-1] = [L(i)..R(i'')-1]$.

Proof of part (b) The transitivity of \bar{A}_Q^Δ and minimality of P implies that if $(Qi, Qj) \in P$ then the previous and the next arcs of P are respectively (i, Qi) and (Qj, j) . The triple of consecutive arcs $(i, Qi), (Qi, Qj), (Qj, j)$ will be denoted as (i, j) , and we will refer to it also as an “arc”. Let $P_Q = (i_1, j_1), \dots, (i_d, j_d)$ be the sequence of all such arcs of P (given in the order that they appear in P). Due to the minimality of P all $2d$ nodes involved must be distinct. It suffices to prove the proposition in the case when $z_i \geq z_j$ for all arcs (i, j) in this sequence. Indeed, if there are arcs (i, j) with $z_i < z_j$ then we can push flow through them afterwards - as discussed in section 5.2, this cannot violate the base polyhedron constraint.

We thus assume that $z_i \geq z_j$ for arcs $(i, j) \in P_Q$. Let (i, j) and (i', j') be two consecutive arcs in the sequence. We claim that $z_j > z_{i'}$. Indeed, since path P is minimal, arc (Qi, Qj') is not in \bar{A}_Q^Δ . If $z_{i'} > z_j$ then $(Qj, Qi') \in \bar{A}_Q^\Delta$, so by transitivity we have $(Qi, Qj') \in \bar{A}_Q^\Delta$ - contradiction. If $z_{i'} = z_j$ then $(Qi, Qi') \in \bar{A}_Q^\Delta$ (since $(Qi, Qj) \in \bar{A}_Q^\Delta$ and $z_{i'} = z_j$), so by transitivity we have $(Qi, Qj') \in \bar{A}_Q^\Delta$ - contradiction.

We showed that $z_{i_1} \geq z_{j_1} > \dots > z_{i_d} \geq z_{j_d}$. This implies that $L(i_1) < R(j_1) < \dots < L(i_d) < R(j_d)$. Now consider $k \in [0..m]$; we need to show that $\hat{g}(k) = g(k) - \sum_{k'=1}^k \hat{z}^k \geq 0$ where \hat{z} is the vector after sending $\lceil \Delta \rceil$ units of flow through P and $\hat{g}(\cdot)$ is the corresponding residual function.

It follows from the definition of z^k that Q can be partitioned into two disjoint subsets S, T with k and $m - k$ nodes, respectively, such that $z_i \geq z^k \geq z_j$ for any $i \in S, j \in T$. Let us introduce the following terminology. Arc (i, j) in P_Q will be called *left-exterior* if $z_i \geq z_j \geq z^k + \lceil \Delta \rceil$ (and thus $i, j \in S$), and *right-exterior* if $z^k - \lceil \Delta \rceil \geq z_i \geq z_j$ (and thus $i, j \in T$). Clearly, after the update we have $\hat{z}_i > \hat{z}_j \geq z^k$ for left-exterior arcs (i, j) and $z^k \geq \hat{z}_i > \hat{z}_j$ for right-exterior arcs (i, j) . An arc in P_Q is called *exterior* if it is either left-exterior or right-exterior, and *interior* otherwise. Note that an interior arc (i, j) must satisfy $z_i \geq z^k \geq z_j$, which is equivalent to the condition $k \in [L(i)..R(j)]$. This implies that P_Q can have at most one interior arc.

We now consider three possible cases.

- All arcs in P_Q are exterior. Then after the update we have $\hat{z}_i \geq z^k \geq \hat{z}_j$ for any $i \in S, j \in T$, so S contains k nodes i with the largest values of \hat{z}_i . This implies that $\hat{g}(k) = g(k) - \hat{z}(S)$. Since each arc (i, j) in P_Q either has both endpoints in S or both endpoints in T , we have $\hat{z}(S) = z(S)$, so $\hat{g}(k) = \bar{g}(k) \geq 0$.
- P_Q has an interior arc (u, v) with $k \in [L(u)..R(v) - 1]$; thus, $\bar{g}(k) \geq 3\Delta/2$ since $(Qu, Qv) \in \bar{A}_Q^\Delta$. We can assume without loss of generality that $u \in S$ and $v \in T$. (Sets S and T could have been chosen in this way since $L(u) \leq k$ and $R(v) > k$). After the update we have $\hat{z}_i \geq z^k \geq \hat{z}_j$ for any $i \in S, j \in T$, so S contains k nodes i with the largest values of \hat{z}_i . This implies that $\hat{g}(k) = g(k) - \hat{z}(S)$. Arc (u, v) is the only one in the sequence which has exactly one endpoint (namely u) in S . Therefore, $\hat{z}(S) = z(S) + \lceil \Delta \rceil$ (where “ $+\lceil \Delta \rceil$ ” term comes from the update $\hat{z}_u = z_u + \lceil \Delta \rceil$), so $\hat{g}(k) = \bar{g}(k) - \lceil \Delta \rceil \geq 3\Delta/2 - \lceil \Delta \rceil \geq 0$.
- P_Q has an interior arc (u, v) with $R(v) = k$. We must have $u, v \in S$ and $z_u \geq z_v = z^k$. After the update we have $\hat{z}_v = z^k - \lceil \Delta \rceil$ and $\hat{z}_i \geq z^k \geq \hat{z}_j$ for any $i \in S - \{v\}, j \in T$. Let (u', v') be the arc in P_Q that immediately follows (u, v) ; if (u, v) is the last arc in P_Q then we say that (u', v') does not exist. Two cases are possible:

- Arc (u', v') does not exist or $z^k - 2\lceil \Delta \rceil \geq z_{u'}$. Then $\hat{z}_v = z^k - \lceil \Delta \rceil \geq z_j$ for any $j \in T$. Thus, S contains k nodes i with the largest values of \hat{z}_i . This implies that $\hat{g}(k) = g(k) - \hat{z}(S)$. Since each arc (i, j) in P_Q either has both endpoints in S or both endpoints in T , we have $\hat{z}(S) = z(S)$, so $\hat{g}(k) = \bar{g}(k) \geq 0$.
- Arc (u', v') exists and $z_{u'} = z^k - \lceil \Delta \rceil$; thus, $L(u') = R(v) + 1 = k + 1$, $z^{k+1} = z^k - \lceil \Delta \rceil$. After the update $\hat{z}_v = z^k - \lceil \Delta \rceil$, $\hat{z}_{u'} = z^k$, so the set $S' = (S - \{v\}) \cup \{u'\}$ contains k nodes i with the largest values of \hat{z}_i . This implies that $\hat{g}(k) = g(k) - \hat{z}(S')$. We have $\hat{z}(S') = \hat{z}(S) + [\hat{z}_{u'} - \hat{z}_v] = z(S) + \lceil \Delta \rceil$, so $\hat{g}(k) = \bar{g}(k) - \lceil \Delta \rceil$. We now need to show that $\bar{g}(k) \geq \lceil \Delta \rceil$.

Conditions $(Qu, Qv), (Qu', Qv') \in \bar{A}_Q^\Delta$ imply that $\bar{g}(k - 1) \geq \lceil 3\Delta/2 \rceil$ and $\bar{g}(k + 1) \geq \lceil 3\Delta/2 \rceil$. We can write

$$\begin{aligned} \bar{g}(k) - \bar{g}(k - 1) &= [g(k) - g(k - 1)] - z^k \\ \bar{g}(k + 1) - \bar{g}(k) &= [g(k + 1) - g(k)] - z^{k+1} \end{aligned}$$

Since $g(\cdot)$ is concave, we have $g(k) - g(k-1) \geq g(k+1) - g(k)$. Thus,

$$\bar{g}(k) - \bar{g}(k-1) + z^k \geq \bar{g}(k+1) - \bar{g}(k) + z^{k+1}$$

$$\begin{aligned} 2\bar{g}(k) &\geq \bar{g}(k-1) + \bar{g}(k+1) - [z^k - z^{k+1}] \\ &\geq \lceil 3\Delta/2 \rceil + \lceil 3\Delta/2 \rceil - \lceil \Delta \rceil \geq 3\lceil \Delta \rceil - \lceil \Delta \rceil = 2\lceil \Delta \rceil \end{aligned}$$

Proof of part (c) Let S be the set used in eq. (14), and denote $T = Q - S$, $k = |S|$. We assume that $S \neq \emptyset$ and $S \neq Q$, otherwise the LHS in eq. (14) would be 0. Let i be a node in S with the minimum value of z_i and j be a node in T with the maximum value of z_j . Since there was no augmenting path upon termination of the previous 2Δ -phase, set $\bar{A}_Q^{2\Delta}$ cannot have arc (Qi, Qj) . Therefore, $z_i \geq z_j$ and $\bar{g}(\bar{k}) \leq \lceil 3\Delta \rceil$ for some $\bar{k} \in [L(i)..R(j) - 1]$. The choice of i, j and condition $z_i \geq z_j$ imply that $\min\{z_{i'} \mid z_{i'} \in S\} \geq \max\{z_{j'} \mid z_{j'} \in T\}$, hence $\bar{f}_Q(S) = \bar{g}(|S|) = \bar{g}(k)$. Thus, we need to show that $\bar{g}(k) \leq \alpha_Q \cdot \lceil \Delta \rceil$ where $\alpha_Q = 3(m-1)$. If $\bar{k} = k$ then the claim is obvious. Suppose that $\bar{k} \neq k$. Two cases are possible:

- $\bar{k} > k$. We have $k+1 < \bar{k}+1 \leq R(j)$, so $z^{k+1} \geq z^{R(j)} = z_j$. We cannot have $z^{k+1} > z_j$ since in this case there would be at least $k+1$ nodes $j' \in Q$ with $z_{j'} > z_j$; by the choice of j these nodes would belong to S , so we would have $|S| \geq k+1$ - contradiction. Thus, we must have $z^{k+1} = z^{k+2} = \dots = z^{R(j)}$. This implies that function $p(k') = \sum_{k''=1}^{k'} z^{k''}$ is linear in $[k..R(j)]$. We have $\bar{g}(k') = g(k') - p(k')$ where $g(\cdot)$ is a concave function, therefore $\bar{g}(\cdot)$ is also concave in $[k..R(j)]$. There holds $\bar{k} \in [k+1..R(j) - 1]$, thus

$$\bar{g}(\bar{k}) \geq \frac{R(j) - \bar{k}}{R(j) - k} \cdot \bar{g}(k) + \frac{\bar{k} - k}{R(j) - k} \cdot \bar{g}(R(j))$$

We have $\bar{g}(\bar{k}) \leq \lceil 3\Delta \rceil$ and $\bar{g}(R(j)) \geq 0$, therefore $\bar{g}(k) \leq \frac{R(j)-k}{R(j)-\bar{k}} \lceil 3\Delta \rceil \leq (m-1) \cdot \lceil 3\Delta \rceil \leq \alpha_Q \cdot \lceil \Delta \rceil$.

- $\bar{k} < k$. We have $L(i) \leq \bar{k} < k$, so $z_i = z^{L(i)} \geq z^k$. We cannot have $z_i > z^k$ since in this case there would be at least $m-k+1$ nodes $i' \in Q$ with $z_{i'} > z_i$; by the choice of i these nodes would belong to T , so we would have $|T| \geq m-k+1$ - contradiction. Thus, we must have $z^{L(i)} = \dots = z^{k-1} = z^k$. This implies that function $p(k') = \sum_{k''=1}^{k'} z^{k''}$ is linear in $[L(i) - 1..k]$. We have $\bar{g}(k') = g(k') - p(k')$ where $g(\cdot)$ is a concave function, therefore $\bar{g}(\cdot)$ is also concave in $[L(i) - 1..k]$. There holds $\bar{k} \in [L(i)..k-1]$, $\bar{g}(\bar{k}) \leq \lceil 3\Delta \rceil$ and $\bar{g}(L(i)-1) \geq 0$, so similar to the previous case we conclude that $\bar{g}(k) \leq \frac{k-(L(i)-1)}{\bar{k}-(L(i)-1)} \lceil 3\Delta \rceil \leq (m-1) \cdot \lceil 3\Delta \rceil \leq \alpha_Q \cdot \lceil \Delta \rceil$.

Appendix D: Implementation of $\text{GetNeighbors}_Q^\Delta(Qi)$ for cardinality-dependent terms

Case 1 Assume that $f_Q(S) = g(|S|)$ for $S \subseteq Q$ where g is concave. We use the same notation as in section 5.2.

First, let us describe the data structure for Q . Nodes $i \in Q$ will be grouped into “supernodes” according to their value of z_i . The set of supernodes is denoted as \tilde{Q} . The cardinality of \tilde{Q} equals the number of unique values in the set $\{z_i \mid i \in Q\}$. At each supernode $u \in \tilde{Q}$ we store values $z_u = z_i$, $L(u) = L(i)$, $R(u) = R(i)$ where i is a node contained in u . We treat supernode u as the set $u = \{Qi \mid i \in Q, z_i = z_u\}$. Supernodes u sorted by their values of z_u will be stored in a doubly-linked list. Each $u \in \tilde{Q}$ also have a pointer to a doubly-linked list of nodes in u , and each

node $Qi \in N$ will have a pointer to $u \in \tilde{Q}$ with $Qi \in u$. Finally, we maintain residual function $\bar{g}(\cdot)$ as an array of size $O(m)$. It is easy to see that after each augmentation this data structure can be dynamically updated in $O(m)$ time.

For each supernode u we maintain flag $\text{REACHED}(u) = \bigwedge_{i \in u} \text{REACHED}(Qi)$; at the beginning of the BFS it is set to **false**. Procedure $\text{GetNeighbors}_{\tilde{Q}}^{\Delta}(Qi)$ is defined as follows:

$\text{GetNeighbors}_{\tilde{Q}}^{\Delta}(Qi)$

- Set $\text{REACHED}(Qi) := \text{true}$ and $S := \emptyset$. Determine supernode u with $Qi \in u$.
- If $\text{REACHED}(u)$ is **true** then stop, otherwise set $\text{REACHED}(u) := \text{true}$ and continue.
- If $\min_{k \in [L(u)..R(u)-1]} \bar{g}(k) \geq 3\Delta/2$ call $\text{Add}(u)$.
- If u has left neighbor u^- with $z_{u^-} > z_u$ call $\text{Add}(u^-)$ and $\text{ProcessLeft}(u^-)$.
- If u has right neighbor u^+ with $z_u > z_{u^+}$ and $\min_{k \in [L(u)..R(u^+)-1]} \bar{g}(k) \geq 3\Delta/2$ call $\text{Add}(u^+)$ and $\text{ProcessRight}(u^+)$.

$\text{ProcessLeft}(u)$

- If $\text{REACHED}(u)$ is **true** then stop, otherwise set $\text{REACHED}(u) := \text{true}$ and continue.
- If u has left neighbor u^- with $z_{u^-} > z_u$ call $\text{Add}(u^-)$ and $\text{ProcessLeft}(u^-)$.

$\text{ProcessRight}(u)$

- If $\text{REACHED}(u)$ is **true** then stop, otherwise set $\text{REACHED}(u) := \text{true}$ and continue.
- If u has right neighbor u^+ with $z_u > z_{u^+}$ and $\min_{k \in [L(u)..R(u^+)-1]} \bar{g}(k) \geq 3\Delta/2$ call $\text{Add}(u^+)$ and $\text{ProcessRight}(u^+)$.

$\text{Add}(u)$

- For each node $Qi \in u$ with $\text{REACHED}(Qi) = \text{false}$ set $\text{REACHED}(Qi) := \text{true}$ and add Qi to S .

The correctness of this procedure should be clear. Note, $\text{ProcessLeft}(u)$ and $\text{ProcessRight}(u)$ are only called when some node $Qi \in u$ has been reached by BFS. If $\text{REACHED}(u)$ is true then all nodes that can be reached from Qi (and from other nodes in u) via arcs in $\bar{A}_{\tilde{Q}}^{\Delta}$ have already been added, which justifies statement “If $\text{REACHED}(u)$ is **true** then stop”. Steps following this statement will be executed at most once for each supernode u , therefore each node, supernode and element of array $\bar{g}(\cdot)$ is accessed at most constant number of times during a single BFS search. Thus, $\beta_Q = O(m)$.

Case 2 We now assume that $f_Q(S) = g(|S \cap Q'|, |S \cap Q''|)$ for $S \subseteq Q$ where Q', Q'' are disjoint subsets of Q . Without loss of generality we can assume that $Q = Q' \cup Q''$. Denote $m' = |Q'|$, $m'' = |Q''|$, $m = |Q| = m' + m''$. Let $y \in \mathbb{R}^{Q'}$ and $z \in \mathbb{R}^{Q''}$ be vectors with $y_i = \varphi_{Qi}$ for $i \in Q'$ and $z_j = \varphi_{Qj}$ for $j \in Q''$. We define sequences $(y^1, \dots, y^{m'})$ and $(z^1, \dots, z^{m''})$ similar to the case above; y^k and z^k are the k -th largest numbers among values y_i and z_i , respectively. Indexes $L(i)$ and $R(i)$ are also defined as in section 5.2; we have $1 \leq L(i) \leq R(i) \leq m'$ for $i \in Q'$ and $1 \leq L(j) \leq R(j) \leq m''$ for $j \in Q''$. Let $\bar{g}(k', k'') = \min\{f_Q(S) \mid |S \cap Q'| = k', |S \cap Q''| = k''\}$ be the “residual function”. We have

$$\bar{g}(k', k'') = g(k', k'') - \sum_{k=1}^{k'} y^k - \sum_{k=1}^{k''} z^k$$

It can be seen that $\bar{g}(\cdot, \cdot)$ is a *Monge matrix* [4], i.e. for any $0 \leq k'_1 < k'_2 \leq m'$ and $0 \leq k''_1 < k''_2 \leq m''$ there holds $\bar{g}(k'_1, k''_1) + \bar{g}(k'_2, k''_2) \leq \bar{g}(k'_1, k''_2) + \bar{g}(k'_2, k''_1)$. This follows from $\bar{f}_Q(S' \cap S'') + \bar{f}_Q(S' \cup S'') \leq \bar{f}_Q(S') + \bar{f}_Q(S'')$ where S' contains first k'_1 nodes of Q' and first k'_2 nodes of Q'' , and S'' contains first k'_2 nodes of Q' and first k'_1 nodes of Q'' . (We assume that nodes in Q' and Q'' are sorted so that components y_i and z_i are non-increasing.) For a row $k' \in [0..m']$ let $k''(k') \in [0..m'']$ be the column that contains the leftmost minimum entry in row k' . Thus, $k''(k') = \min\{k'' \in [0..m''] \mid \bar{g}(k', k'') = \min_{k''} \bar{g}(k', k'')\}$. It is known [4] that Monge matrices are *monotone*, i.e. $k''(0) \leq k''(1) \leq \dots \leq k''(m')$. Furthermore, they are *totally monotone*, i.e. every submatrix is monotone. As shown by Aggarwal et al. [1], indexes $k''(0), \dots, k''(m')$ for a totally monotone matrix can be computed in $O(m)$ time.

We can describe data structures for implementing $\text{GetNeighbors}_Q^\Delta(Qi)$. Nodes in $i \in Q'$ will be grouped into supernodes according to the values y_i analogously to case 1. A similar data structure will be used for nodes in Q'' . We will maintain an array of cumulative sums $\sum_{k=1}^{k'} y^k$ for $k' \in [0..m']$ and $\sum_{k=1}^{k''} z^k$ for $k'' \in [0..m'']$, which will allow computing $\bar{g}(k', k'')$ in $O(1)$ time. At the beginning of each BFS we will compute indexes $k''(k')$ for $k' \in [0..m']$ using the algorithm in [1] and also indexes $k'(k'') = \min\{k' \in [0..m'] \mid \bar{g}(k', k'') = \min_{k'} \bar{g}(k', k'')\}$ for each column $k'' \in [0..m'']$.

Arcs in \bar{A}_Q^Δ can be split into four groups $\bar{A}_{00}, \bar{A}_{01}, \bar{A}_{10}, \bar{A}_{11}$ where $\bar{A}_{\alpha\beta} = \{(Qi, Qj) \in \bar{A}_Q^\Delta \mid [i \in Q'] = \alpha, [j \in Q''] = \beta\}$ and $[\cdot]$ is the Iverson bracket: it is 1 if its argument is true, and 0 otherwise. Consider the version of $\text{GetNeighbors}_Q^\Delta(Qi)$ that processes only arcs in a specific group, rather than all arcs in \bar{A}_Q^Δ . It suffices to show how to implement such procedure for each of the four groups; these procedures will be called sequentially.

First, consider arcs in \bar{A}_{11} . Using the same argumentation as in section 5.2 we conclude that sending flow from a node Qi to another node Qj ($i, j \in Q'$) is possible if and only if one of the two conditions hold: (a) $y_i < y_j$; (b) $y_i \geq y_j$ and $\min_{k' \in [L(i)..R(j)-1]} \bar{g}'(k') \geq 1$ where we defined $\bar{g}'(k') = \min_{k''} \bar{g}(k', k'')$. Thus, the set \bar{A}_{00} is constructed completely analogously to the set \bar{A}_Q^Δ in section 5.2 except that function \bar{g} is replaced with \bar{g}' and threshold $3\Delta/2$ is replaced with 1. Accordingly, we can use an obvious adaptation of the procedure for the case 1. Note, $\bar{g}'(k')$ can be evaluated in $O(1)$ time using arrays of indexes $k''(k')$ and cumulative sums for vectors y and z . Arcs in \bar{A}_{00} can be handled in a similar way. It remains to show how to handle arcs in \bar{A}_{10} (the set \bar{A}_{01} will follow by symmetry).

Consider nodes $i \in Q', j \in Q''$. Sending $\lceil \Delta \rceil$ units of flow from i to j via Q , i.e. the operation $y_i := y_i + \lceil \Delta \rceil, z_j := z_j - \lceil \Delta \rceil$, affects function $\bar{g}(\cdot, \cdot)$ as follows: values $\bar{g}(k', k'')$ for $(k', k'') \in [L(i)..m'] \times [1..R(j) - 1]$ are decreased by $\lceil \Delta \rceil$ and values $\bar{g}(k', k'')$ for $(k', k'') \in [1..L(i) - 1] \times [R(j)..m'']$ are increased by $\lceil \Delta \rceil$. Thus, sending flow is possible if and only if

$$\min\{\bar{g}(k', k'') \mid (k', k'') \in K(L(i), R(j))\} > 0, \quad K(a, b) = [a..m'] \times [0..b - 1]$$

There holds $K(a, b_1) \subset K(a, b_2)$ for $b_1 < b_2$, therefore the set of arcs in \bar{A}_{10} from Qi have the form $\{(Qi, Qj) \mid R(j) \leq b(L(i))\}$ where

$$b(a) = \max \left\{ b \in [1..m''] \mid \min_{(k', k'') \in K(a, b)} \bar{g}(k', k'') > 0 \right\} \quad a \in [1..m'] \quad (21)$$

(If the set in (21) is empty then we assume that $b(a) = 0$.) We compute indexes $b(a)$ at the beginning of BFS in linear time using the following recursion:

$$\begin{aligned} b(m') &= k''(m') && (\text{since } \min_{k''} \bar{g}(m', k'') = \bar{g}(m', m'') = 0) \\ b(a) &= \begin{cases} b(a+1) & \text{if } \bar{g}(a, k''(a)) > 0 \\ \min\{b(a+1), k''(a)\} & \text{if } \bar{g}(a, k''(a)) = 0 \end{cases} && \forall a \in [1..m' - 1] \end{aligned}$$

Note that $0 \leq b(1) \leq \dots \leq b(m') \leq m''$. Procedure $\text{GetNeighbors}_Q^\Delta(Qi)$, $i \in Q'$ for the set of arcs \bar{A}_{10} is implemented as follows. First, we locate the rightmost supernode $v \subseteq Q''$ satisfying

$R(v) \leq b(L(i))$. (Pointers to these supernodes for each supernode $u \subseteq Q'$ can be computed at the beginning of BFS.) We then call procedure $\text{Add}(v)$, which is defined as in the case 1, and procedure $\text{ProcessLeft}_{10}(v)$ defined as follows:

$\text{ProcessLeft}_{10}(v)$

- If $\text{REACHED}(v)$ is **true** then stop, otherwise set $\text{REACHED}(v) := \text{true}$ and continue.
- If v has left neighbor v^- with $z_{v^-} > z_v$ call $\text{Add}(v^-)$ and $\text{ProcessLeft}_{10}(v^-)$.

References

- [1] A. Aggarwal, M. M. Klawe, S. Moran, P. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2:195–208, 1987.
- [2] A. Billionnet and M. Minoux. Maximizing a supermodular pseudo-boolean function: a polynomial algorithm for supermodular cubic functions. *Discrete Applied Mathematics*, 12(1):1–11, 1985.
- [3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 23(11):1124–1137, September 2004.
- [4] Rainer E. Burkard, Bettina Klinz, and Rüdiger Rudolf. Perspectives of Monge properties in optimization. *Discrete Applied Mathematics*, 70(2):95–161, 1996.
- [5] Martin C. Cooper. Minimization of locally defined submodular functions by optimal soft arc consistency. *Constraints*, 13(4):437–458, 2008.
- [6] Andrew DeLong, Anton Osokin, Hossam Isack, and Yuri Boykov. Fast approximate energy minimization with label costs. In *CVPR*, 2010.
- [7] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications*, volume 17, pages 69–87. Gordon and Breach, 1970.
- [8] Jack Edmonds and Rick Giles. A min-max relation for submodular functions on graphs. *Annals of Discrete Mathematics*, 1:185–204, 1977.
- [9] L. Fleischer, S. Iwata, and S. T. McCormick. A faster capacity scaling algorithm for submodular flow. *Mathematical Programming*, 92:119–139, 2002.
- [10] D. Freedman and P. Drineas. Energy minimization via graph cuts: settling what is possible. In *CVPR*, pages 939–946, 2005.
- [11] S. Fujishige. *Submodular Functions and Optimization*. North-Holland, 1991.
- [12] S. Fujishige and S. Iwata. Minimizing a submodular function arising from a concave function. *Discrete Applied Mathematics*, 92(2-3):211–215, 1999.
- [13] S. Fujishige and X. Zhang. New algorithms for the intersection problem of submodular systems. *Japan J. Indust. Appl. Math.*, 9:369–382, 1992.
- [14] Satoru Fujishige. Algorithms for solving the independent-flow problems. *J. Oper. Res. Soc. Japan*, 21:189–204, 1978.
- [15] Satoru Fujishige and Satoru Iwata. Algorithms for submodular flows. *IEICE Trans. Inf. & Syst.*, E83-D(3):322–329, 2000.
- [16] Andrew V. Goldberg and Satish Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783–797, 1998.
- [17] Dorit S. Hochbaum and Vikas Singh. An efficient algorithm for co-segmentation. In *ICCV*, 2009.
- [18] S. Iwata. A capacity scaling algorithm for convex cost submodular flows. *Math. Programming*, 76(2):299–308, 1997.
- [19] S. Iwata. A faster scaling algorithm for minimizing submodular functions. *SIAM J. Comput.*, 32(4):833–840, 2003.

- [20] Satoru Iwata, S. Thomas McCormick, and Maiko Shigeno. A fast cost scaling algorithm for submodular flow. *Information Processing Letters*, 74(3-4):123–128, 2000.
- [21] Satoru Iwata and James B. Orlin. A simple combinatorial algorithm for submodular function minimization. In *SODA*, pages 1230–1237, 2009.
- [22] P. Kohli, L. Ladicky, and P. Torr. Robust higher order potentials for enforcing label consistency. In *CVPR*, 2008.
- [23] Pushmeet Kohli, M. Pawan Kumar, and Philip H.S. Torr. P^3 & beyond: Move making algorithms for solving higher order functions. *PAMI*, 31(9):1645–1656, 2009.
- [24] James B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.
- [25] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Joint optimization of segmentation and appearance models. In *ICCV*, 2009.
- [26] Stanislav Žitný, David A. Cohen, and Peter G. Jeavons. The expressive power of binary submodular functions. *Discrete Applied Mathematics*, 157(15):3347–3358, 2009.
- [27] Stanislav Žitný and Peter G. Jeavons. Classes of submodular constraints expressible by graph cuts. *Constraints*, 2009.